Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Převodník Excel - OWL

Plzeň 2018 Patrik Jaroš

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 26. června 2018

Patrik Jaroš

Abstract

This bachelor thesis deals with the problem of communication with domain experts in the phase of modeling an ontology describing a particular domain. The aim of the thesis is to create a converter between OWL and Excel files, allowing domain experts to work in Excel environment. In the introduction, the reader familiarizes with RDF, RDF Schema, and OWL. The next part of thesis describes the proposal of ontology data structuring in Excel and the implementation of the converter. In the end, the correct functionality of the converter is verified, with respect to requirements mentioned in the proposal.

Key words Excel, OWL, converter, ontology, RDF

Abstrakt

Tato bakalářská práce se zabývá problémem komunikace s doménovými experty při modelování ontologie určité domény. Cílem práce je vytvořit převodník mezi OWL a Excelovskými soubory, který by umožnil doménovým expertům pracovat v prostředí Excelu. V úvodu práce seznamuje čtenáře s oblastí RDF, RDF Schema a OWL. V další části práce je popsán návrh strukturalizace dat ontologie v Excelu a implementace převodníku. Na konci práce je ověřena správná funkčnost převodníku s ohledem na požadavky zmíněné v návrhu.

Klíčová slova Excel, OWL, převodník, ontologie, RDF

Obsah

Úvod

Znalostní experti pohybující se v různých odvětvích většinou znají Excelovské sešity a zvládají základní práci s nimi, ale ne vždy však znají jazyk OWL. Jazyk OWL (Web Ontology Language) je značkovací jazyk vyvinutý organizací W3C (World Wide Web Consortium), navržený pro reprezentaci komplexních znalostí o určitých věcech, množinách těchto věcí a vztazích mezi nimi.

OWL dokumenty, známé jako ontologie, uchovávají všechny tyto znalosti v textovém formátu a jsou definovány pomocí formálního jazyka. Ontologie jsou využívány ve větší míře ve znalostních aplikacích a jejich účelem je specifikace určité oblasti (domény).

Při modelování znalostní domény pomocí OWL je problematické efektivně komunikovat se znalostními experty. Cílem práce je vytvořit převodník, který by převedl strukturovaně zapsané znalosti z Excelovského sešitu do validní OWL ontologie a zpět. Tento převodník by měl umožnit znalostnímu expertovi pracovat v prostředí Excelu, které je mu často dobře známé a bližší než práce se samotnou ontologií. Převodník by tak odboural technologickou bariéru v komunikaci. Měl by umět i převod opačný, tedy převod existující OWL ontologie do Excelovského sešitu. Tato funkce se využije při spolupráci více lidí nad jednou ontologií, kdy při komunikaci s doménovým expertem bude potřeba převést ontologii do Excelovského sešitu pro úpravu znalostí v ní uložených.

Na začátku této práce popisuji jazyk OWL. Po přečtení by měl čtenář získat představu o tom, co to OWL je a jak tato technologie funguje. Dalším bodem je vyhledání existujících řešení tohoto problému a analýza jejich výhod a nevýhod. Následně je práce zaměřena na návrh způsobu přepisu OWL ontologie do Excelovského sešitu s ohledem na uživatelskou přívětivost a přehlednost pro znalostního experta. V poslední části práce je zdokumentována implementace převodníku, otestování jeho správné funkčnosti a zhodnocení dosažených výsledků.

1 RDF

Předtím, než se budeme věnovat OWL, je třeba se seznámit s RDF. Resource Description Framework (dále jen RDF), je framework pro vyjádření informací o zdrojích. Pod pojmem zdroje (anglicky resources) si lze představit prakticky cokoliv, například lidi, zvířata, fyzické objekty reálného světa, nebo třeba textové dokumenty na počítači. Tento framework spravuje organizace World Wide Web Consortium (W3C).[1] Organizace W3C stojí za vývojem webových standardů, například HTML nebo CSS. [1]

RDF se používá v situacích, kde informace a znalosti nejsou pouze zobrazovány uživatelem, ale jsou převážně zpracovávány aplikacemi. Zejména se RDF používá pro zveřejnění informací a jejich propojení na webu. Například na odkazu *http://www.example.org/bob#me* můžeme najít informace o osobě jménem Bob, včetně informace o tom, že se zná s Alicí, která je taktéž identifikována svým odkazem URI (Uniform Resource Identifier, více o URI v sekci 1.1.1). Pomocí tohoto identifikátoru můžeme následně zjistit informace o Alici. [1]

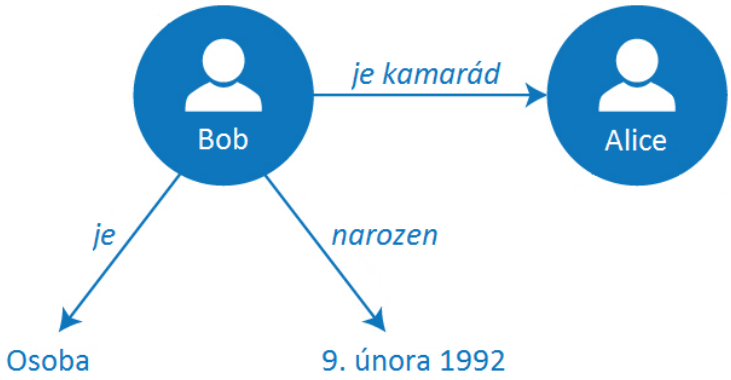
V praxi však URI mohou sloužit i jako identifikátory zdroje, které neodkazují na žádnou webovou stránku, či soubor s informacemi o zdroji.

1.1 Datový model RDF

Jak již bylo řečeno, RDF nám umožňuje vyjádřit informace o nějakých zdrojích. Tyto informace jsou vyjádřeny pomocí RDF statementů (do češtiny lze přeložit jako tvrzení), které jsou základní strukturou RDF. Formát těchto tvrzení je jednoduchý, vždy má strukturu: <subjekt> <predikát> <objekt>. [1]

Protože RDF tvrzení obsahují tři prvky, říká se jim RDF trojice. Tyto trojice mohou vyjadřovat vztah dvou zdrojů, případně vztah zdroje a hodnoty. Subjekt vždy představuje jednoznačně identifikovatelný zdroj, který je ve vztahu s objektem. Objekt může také představovat nějaký zdroj, nebo hodnotu s definovaným datovým typem (například řetězec, nebo celé číslo). V případě, že je objekt vyjádřen hodnotou, jedná se o literál. Bližší seznámení s literály se nachází v sekci 1.1.2.

[1] https://www.w3.org/



Obrázek 1.1: Zobrazení trojic v grafu

Predikát je prvkem trojice, který vyjadřuje typ vztahu subjektu s objektem a platí pouze v jednom směru. Například trojice <Bob> <je> <Osoba> vyjadřuje pouze to, že Bob je osobou. Logicky však nemůže platit, že každá osoba je Bob. [1]

Jednotlivé zdroje mohou být součástí více trojic. Bob, který je osobou, může mít také přiřazené datum narození. Zdroj může být v jedné trojici subjektem a v jiné objektem. Díky této vlastnosti je možné mezi sebou jednotlivé trojice propojit. Propojování trojic je podstatnou součástí modelování v RDF. Je také využíváno v Linked Open Data, kde je to považováno za nejlepší formát, ve kterém lze otevřená data držet [2].

Seskupení trojic můžeme vizualizovat pomocí grafů, které se skládají z uzlů a hran. Uzly představují subjekty a objekty, orientované hrany grafů představují predikáty. [1]

V obrázku 1.1 se nachází graf, složený ze tří trojic zmíněných v předchozím textu.

1.1.1 URI

Uniform Resource Identifier (dále jen URI) je jednoznačný identifikátor zdroje. URL (Uniform Resource Locator) které lidé používají jako webové adresy, jsou jednou z forem URI. Další formou URI je identifikace pro nějaký zdroj, bez předání jeho umístění nebo toho, jak ke zdroji přistoupit.

URI mohou být obecně použity pro jakýkoliv prvek trojice. Jedná se o globální identifikátory, je tedy bez problémů možné je opakovaně použít pro označení jedné a té samé věci. Identifikátorům může být také přiřazen určitý význam pomocí slovníků nebo konvencí. Více o slovnících v sekci 1.3.

K identifikaci zdroje lze také použít IRI (Internationalized Resource Iden-

tifier), který funguje obdobně. Výhodou IRI oproti URI je to, že v řetězci identifikátoru IRI lze použít i znaky mimo ASCII tabulku, například tedy speciální české znaky, nebo znaky z jiných jazyků [1]. Jedná se však zároveň o nevýhodu, kvůli možnosti phishingových útoků [3].

1.1.2 Literály

Literály jsou hodnoty s přiřazeným datovým typem. Typickým příkladem literálu je řetězec, datum, nebo celé číslo. Řetězcům může být navíc přiřazeno kódové označení jazyku řetězce. V trojici mohou být literály použity pouze v pozici objektu.

1.1.3 Prázdné uzly

URI a literály jsou základními stavebními kameny RDF tvrzení. Mimo to se nám občas můžou hodit prázdné uzly, které si lze jednoduše představit jako proměnné, reprezentující věc, jejíž ID (název) neznáme. Prázdné uzly nemají globální identifikátor. V RDF je lze použít v pozici subjektu i objektu, můžeme jim tedy přiřadit nějaká data a vlastnosti.

1.1.4 Grafy

Grafem nazýváme množinu trojic v RDF modelu. Příklad takového grafu byl na obrázku 1.1. V tomto grafu jsou 3 trojice. Více takových grafů tvoří datovou sadu RDF, kde každý graf má své jméno. Jméno grafu představuje identifikátor URI, který je grafu přiřazen. V datové sadě se může také vyskytovat nejvýše 1 bezejmenný (default) graf. Každá trojice v datové sadě má přesně určený graf, ve kterém se nachází, nemůže se tak nacházet ve více grafech. Přiřazení jména grafu k trojici vede k vytvoření tzv. RDF čtveřic (quads), ve kterých se kromě subjektu, predikátu a objektu nachází ještě identifikátor (URI) grafu. [1]

1.2 Syntaxe RDF grafů

Existuje několik různých formátů pro zápis RDF grafů. Jiný formát zápisu určitého grafu ale vždy vede k vytvoření stejných trojic, syntaxe jsou tedy logicky ekvivalentní. V této sekci si krátce představíme tři základní syntaxe, N-Triples, Turtle a RDF/XML. U každé syntaxe bude příklad zápisu dvou trojic pro jejich porovnání. První trojicí je tvrzení, že Bob je osoba (Person). Druhé tvrzení přiřazuje Bobovi datum narození.

1.2.1 N-Triples

Jedná se o nejjednodušší syntaxi pro zápis RDF grafu. Na každém řádku je jedna trojice ukončená tečkou. Začátek URI je označen znakem „<“, jeho konec znakem „>“. Přiřazení datového typu k literálu se provede s pomocí oddělovače „^^“. Vzhledem k absenci jakýchkoliv syntaktických zkratek a zjednodušení bude výsledný soubor veliký. Výhodou syntaxe N-Triples je její jednoduchost a to, že se soubor dobře parsuje. Příklad syntaxe N-Triples je ve výpisu 1.1.

```
<http://example.org/bob#me> <http://www.w3.org
    /1999/22-rdf-syntax-ns#type> <http://xmlns.com/foaf
    /0.1/Person> .
<http://example.org/bob#me> <http://schema.org/
    birthDate> "1992-02-09"^^<http://www.w3.org/2001/
    XMLSchema#date> .
```

Výpis 1.1: Příklad syntaxe N-Triples

1.2.2 Turtle

Syntaxe Turtle je rozšířením N-Triples. K základní N-Triples syntaxi přidává syntaktické zkratky, například pro zápis URI pomocí prefixů, nebo pro relativní URI vzhledem k bázi. V příkladu Turtle syntaxe (1.2) jsou na prvních řádcích definovány zkratky. Relativní URI (např. `<bob#me>`) se vztahují k bázovému (`base`) URI. Prefixy slouží k zápisu prefixovaných jmen zdrojů (např. `foaf:Person`), místo URI v plné délce.

```
@base <http://example.org/>
@prefix rdf: <http://www.w3.org/22-rdf-syntax-ns#>
@prefix foaf: <http://xmlns.com/foaf/0.1/>
@prefix schema: <http://schema.org/>
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>

<bob#me> rdf:type foaf:Person ;
schema:birthDate "1992-02-09"^^xsd:date .
```

Výpis 1.2: Příklad syntaxe Turtle

Další výhodou Turtle syntaxe je zkratka pro zápis množiny trojic se stejným subjektem, v našem případě je subjektem Bob. Jednotlivé trojice jsou oddělené středníkem, za poslední trojicí je tečka. Podobně lze také zkrátit zápis přiřazení více objektů pro trojice se stejným subjektem i predikátem. V takovém případě se objekty oddělují čárkou. Tyto syntaktické zkratky

mají za následek to, že stejný kód při větším počtu trojic v syntaxi Turtle je o poznání kratší. To vede k úspoře místa na disku, navíc je kód se zkratkami pro uživatele čitelnější.

1.2.3 RDF/XML

Syntaxe RDF/XML je založena na značkovacím jazyce XML[2] (Extensible Markup Language). Jedná se o vůbec první syntaxi RDF, která se používá již od samotného vývoje RDF v 90. letech minulého století. Ukázka syntaxe RDF/XML se nachází ve výpisu 1.3.

Při definování RDF grafu v RDF/XML syntaxi jsou trojice ohraničeny elementem `rdf:RDF`. Atributy elementu `rdf:RDF` začínající klíčovým slovem `xmlns`, slouží k definici prefixů jmenných prostorů, podobně jako u syntaxe Turtle. Element `rdf:Description` ohraničuje sadu trojic se společným zdrojem na pozici subjektu, zdroj je definován svým URI v parametru `rdf:about` ohraničujícího elementu. [1]

Jednotlivé trojice ohraničené sady jsou reprezentovány jako sub-elementy. Názvem každého sub-elementu je URI, představující predikát trojice. Pokud se i v pozici objektu trojice nachází zdroj definovaný svým URI, sub-element nemá žádný obsah a URI objektu je uloženo v atributu `rdf:resource` sub-elementu. Pokud je však objektem trojice literál, URI datového typu je uloženo v atributu `rdf:datatype` a hodnota literálu je obsahem sub-elementu trojice. Není-li datový typ literálu explicitně vyjádřen, jedná se o obyčejný řetězec znaků. [1]

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:schema="http://schema.org/">

<rdf:Description rdf:about="http://example.org/bob#me">
<rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person
    "/>
<schema:birthDate rdf:datatype="http://www.w3.org/
    XMLSchema#date">1990-07-04</schema:birthDate>
</rdf:Description>
</rdf:RDF>
```

Výpis 1.3: Příklad syntaxe RDF/XML

Jak je vidět v ukázce 1.3, RDF/XML používá podobné syntaktické zkratky jako syntaxe Turtle, nevyužívá je ale v takové míře a kód v této syntaxi je

[2] https://www.w3.org/XML/

o poznání delší. Na rozdíl od syntaxe Turtle jsou prefixy v RDF/XML použitelné pouze ve jménech elementů a jejich atributů. Výhodou této syntaxe je lehčí parsování v porovnání s Turtle.

1.3 Slovníky

Datový model RDF umožňuje vytvářet tvrzení o zdrojích, v modelu ale není nijak zachyceno, co jednotlivá URI predikátů (vlastností) představují a jaký mají význam. V praxi se proto RDF používá v kombinaci se slovníky, které nám umožní definovat sémantické (významové) informace o jednotlivých zdrojích. Pro definici slovníků nabízí RDF rozšíření RDF Schema. [1]

RDF Schema je sémantickým rozšířením RDF. Jedná se o slovník, který poskytuje mechanismy pro popis skupin souvisejících zdrojů a vztahů mezi těmito zdroji. RDF Schema využívá systém tříd a vlastností, který je podobný objektově orientovanému programování a používá se k vytváření datových modelů a ontologií. Více informací o ontologiích se nachází v kapitole 2.

RDF Schema se skládá z množiny pojmů, které jsou využívány pro definici zdrojů ve slovnících. Tyto pojmy se nachází ve dvou jmenných prostorech. Prvním z nich, obvykle s prefixem *rdfs:*, je jmenný prostor identifikovaný URI *http://www.w3.org/2000/01/rdf-schema#*. Druhým jmenným prostorem s prefixem *rdf:* je jmenný prostor *http://www.w3.org/1999/02/22-rdf-syntax-ns#*. Pro přehlednost a lepší čitelnost budou v tomto textu použity zmíněné prefixy. Podle specifikace [4] se pojmy RDF Schema dělí na třídy a vlastnosti. Třídami jsou následující pojmy:

rdfs:Class – hlavní třída, jejími instancemi jsou všechny definované třídy zdrojů.

rdfs:Resource – třída všech zdrojů definovaných v RDF. Všechny zdroje jsou instancemi této třídy.

rdfs:Literal – třída všech literálů. Je instancí *rdfs:Class* a podtřídou *rdfs:Resource*.

rdfs:Datatype – třída datových typů literálů. Je instancí i podtřídou třídy *rdfs:Class*. Každá instance této třídy je podtřídou *rdfs:Literal*.

rdf:langString – třída řetězců s kódovým označením jazyka. Je instancí *rdfs:Datatype* a podtřídou *rdfs:Literal*.

rdf:Property – třída všech RDF vlastností. Je to instance třídy *rdfs:Class*.

Vlastností nazýváme vztah mezi zdroji – subjektem a objektem. Všechny vlastnosti RDF Schema jsou instancemi třídy *rdf:Property*. Zde je seznam těchto vlastností:

rdf:type – slouží k přiřazení zdroje do nějaké třídy.

rdfs:range – umožňuje definovat, z jaké třídy bude zdroj na pozici objektu nějaké vlastnosti. Podobné oboru hodnot v matematice.

rdfs:domain – pro definici, z jaké třídy bude zdroj na pozici subjektu nějaké vlastnosti. Podobné definičnímu oboru v matematice.

rdfs:subClassOf – k vyjádření vztahu mezi dvěma třídami, kdy třída na pozici subjektu trojice je podtřídou třídy na pozici objektu trojice.

rdfs:subPropertyOf – vlastnost pomocí které lze definovat, že jedna vlastnost je podřízena druhé vlastnosti. Pokud budou dva prvky v relaci pomocí podřízené vlastnosti, budou automaticky v relaci vlastnosti nadřízené.

rdfs:label a rdfs:comment – vlastnosti slouží k přiřazení lidsky čitelného popisu zdroje ve formě řetězce.

V RDF Schema jsou definovány i jiné třídy a vlastnosti, například pro práci s kontejnery (množinami prvků), nebo pro popis jednotlivých RDF trojic [4]. Nicméně pro tuto práci nejsou stěžejní, nebudeme je tedy dopodrobna probírat.

Výhodou slovníků je možnost opětovného využívání jejich prvků. Je strojově pohodlnější a fakticky přesnější využívat již nadefinovanou vlastnost s určitým sémantickým významem, než pro daný účel vytvářet vlastnost novou a následně definovat její ekvivalenci s nadefinovanou vlastností ze slovníku. Například pro popisek zdroje využijeme vlastnost *rdfs:label* ze slovníku RDF Schema.

Dalším podstatným nástrojem při vytváření ontologií a komplexnějším modelování dat je jazyk OWL. OWL je také RDF slovníkem a obvykle se používá v kombinaci s RDF Schema. Více o OWL v kapitole 2.

2 OWL

Web Ontology Language (dále jen OWL) je podle [5] deskriptivní (popisný), nikoliv programovací, jazyk pro reprezentaci komplexních znalostí a vytváření ontologií. Momentálně je ve verzi 2 z roku 2012.

Ontologií nazýváme množinu popisných tvrzení o určité části světa (doméně). Při vytváření ontologie se využívá datového modelu RDF a lze ji zobrazovat jako graf. Pro popis prvků z domény a jejich vzájemných vztahů se využívá termínů slovníku RDF Schema. RDF Schema však nabízí pouze jednoduché konstrukce zmíněné v sekci 1.3. Oproti tomu OWL disponuje komplexnějšími nástroji, které si představíme v této kapitole. Nástroje se nachází ve jmenném prostoru *http://www.w3.org/2002/07/owl#* s prefixem *owl:* [5]. Ještě předtím si ale zavedeme pár základních pojmů:

Axiom – obecné tvrzení, které je obsaženo v ontologii.

Třída (class) – množina zdrojů s podobnými vlastnostmi, například osob nebo zvířat.

Jedinec (individual) – jeden konkrétní objekt, který může být instancí jedné, nebo více tříd. Například osoba Alice může být jedincem třídy Člověk a Žena.

Objektová vlastnost (object property) – vyjadřuje vztah (relaci) dvou jedinců, například manželství Boba a Alice.

Datová vlastnost (datatype property) – přiřazuje jedinci hodnotu (literál), například Bobovi datum narození.

Anotační vlastnost (annotation property) – umožňuje přidat další popisné informace k prvkům ontologie, případně k ontologii samotné, ve formě metadat. Můžeme například přidat informaci o tom, kdo je autorem ontologie, nebo jakou doménou se ontologie zabývá.

Nyní se blíže podíváme na detaily modelování v OWL. Uvedeny budou základní nástroje, které OWL nabízí, včetně příkladů. Následně přejdeme ke složitějším konstrukcím. V příkladech bude využita syntaxe Turtle.

2.1 Základní nástroje OWL

Při tvorbě ontologie, se v první řadě provádí tzv. deklarace entit – je třeba zmínit všechny nové třídy, jedince a vlastnosti, které budou v ontologii pou-

žity. V případě potřeby lze entity importovat z jiné ontologie, ve které byly deklarovány. Deklarace entit docílíme využitím konstrukce slovníku RDF Schema, vlastností *rdf:type*, kterou lze využít také pro přiřazení jedince k libovolné třídě. Za zmínku ještě stojí konstrukce *rdfs:subClassOf*, která nám pomůže při vytváření hierarchie tříd, nebo *rdfs:domain* a *rdfs:range* pro omezení definičního oboru a oboru hodnot vlastností.

K tomu OWL přidává možnost definice ekvivalentních a disjuktních tříd. Ekvivalentní třídy jsou takové, které obsahují úplně stejnou množinu jedinců. Disjunktní třídy naopak nemají žádného společného jedince. Propojení jedinců mezi sebou se provádí pomocí objektových vlastností. [5]

Příklad využití zmíněných prvků můžete vidět ve výpisu 2.1, kde se kromě definice prázdného prefixu nachází deklarace entit jedince `Bob`, třídy `Person` a vlastnosti `hasWife`, definice podtřídy `Woman`, omezení definičního oboru a oboru hodnot vlastnosti, nakonec pak propojení dvou jedinců objektovou vlastností.

```
@prefix : <http://example.com/ontology.owl#> .

:Bob      rdf:type owl:NamedIndividual .
:Person   rdf:type owl:Class .
:hasWife  rdf:type owl:ObjectProperty .

:Woman rdfs:subClassOf :Person .

:hasWife  rdfs:domain :Man ;
          rdfs:range  :Woman .

:Bob :hasWife :Alice .
```

Výpis 2.1: Příklad použití základních prvků OWL

2.2 Komplexní třídy

V předchozích sekcích pro nás třída představovala nějaký pojmenovaný typ objektu, kterému byla přiřazena množina jedinců. Takové třídy jsou tzv. atomické. Nyní si ukážeme třídy komplexní. To jsou třídy definované s pomocí již existujících tříd, jedinců a vlastností. K tomu OWL poskytuje třídní konstruktory, využívající například klasické množinové operace – průnik (intersection), sjednocení (union) a doplněk (complement), případně jejich kombinaci. [5]

Ve výpisu 2.2 se nachází konstruktor komplexní třídy matek (*Mother*), jenž je vytvořena z průniku tříd žen (*Woman*) a rodičů (*Parent*). Výsledkem bude třída obsahující všechny jedince, kteří jsou instancemi obou tříd zároveň.

```
:Mother   owl:equivalentClass   [
    rdf:type                owl:Class ;
    owl:intersectionOf   ( :Woman :Parent )
  ] .
```

Výpis 2.2: Vytvoření nové komplexní třídy průnikem jiných tříd

Mimo to můžeme v OWL při vytváření komplexních tříd využít omezení zahrnující vlastnosti (vztahy). V takovém případě můžeme třeba pomocí existenčního omezení definovat instance třídy jako objekty, které jsou v konkrétním vztahu s alespoň jedním, nebo více objekty z konkrétní třídy. Využít lze také omezení, kdy všechny instance nově definované třídy musí být ve vztahu s právě jedním konkrétním jedincem. Ke třídě můžeme navíc přiřadit množinu (datových nebo objektových) vlastností, jejichž hodnoty budou představovat tzv. klíč sloužící k jednoznačné identifikaci instancí třídy. Příkladem klíče pro instance třídy osob by mohlo být rodné číslo osoby.

Posledním omezením je pak kardinalitní omezení. Pomocí něj můžeme vyčíslit počet objektů přiřazených určitou vlastností pro jednoho jedince vytvářené třídy. Všechna omezení jsou detailně popsána ve specifikaci [5].

2.3 Pokročilé modelování vlastností

V sekci 2.2 jsme používali vlastnosti při vytváření nových komplexních tříd. Nyní se podíváme na možnosti modelování vlastností samotných. V základním datovém modelu RDF představuje vlastnost orientovanou hranu grafu, vlastnost tedy platí pouze v jednom směru. Toho lze v OWL využít při vytváření inverzní vlastnosti s podobným významem.

Objektová vlastnost je vztahem mezi dvěma jedinci. Představuje v podstatě binární relaci, kterou známe z matematiky. OWL nabízí modifikaci vlastností podle druhů binárních relací. Je možné definovat vlastnost jako symetrickou, asymetrickou, případně reflexivní. Další možností je definice tzv. funkční vlastnosti, pro kterou platí, že subjekt je vlastností propojen s nanejvýš jedním objektem. [5]

Ve výpisu 2.3 je znázorněno použití definice inverzní vlastnosti (*hasParent*), symetrické vlastnosti (*hasFriend*) a definování funkční vlastnosti (*hasHusband*).

```
:hasParent owl:inverseOf :hasChild .
:hasFriend rdf:type owl:SymmetricProperty .
:hasHusband rdf:type owl:FunctionalProperty .
```

Výpis 2.3: Vlastnosti a jejich modelování

Po zapsání informací a znalostí do ontologie je možné pomocí speciálního softwarového vybavení s ontologiemi dále pracovat. K vytváření a úpravě ontologií v přehledném uživatelském rozhraní slouží editory ontologií (například Protégé[1], který jsem během své práce využíval). Dalším užitečným softwarem je tzv. reasoner, pomocí nějž se dají odvodit znalosti, které nejsou v ontologii explicitně vyjádřeny. Reasoner také dokáže odhalit chyby v konzistenci dat vyhledáním logických sporů. [5]

[1] https://protege.stanford.edu/

3 Existující řešení

Před samotným návrhem a implementací vlastního převodníku jsem se pokusil vyhledat nějaký program, který by prováděl námi požadovanou činnost, tedy převod OWL ontologie do Excelovského sešitu a zpět. Při svém hledání jsem narazil na dvě podobná řešení, žádné z nich však plně nevyhovuje našim požadavkům. Na základě toho jsem se po dohodě s vedoucím práce rozhodl vytvořit vlastní nové řešení převodníku.

Nalezená řešení jsem v průběhu práce blíže zkoumal a v této kapitole je stručně popisuji. Prvním z programů je komerční TopBraid Composer[1] od společnosti TopQuadrant[2], druhým je pak open source nástroj Populous[3] vytvořený projektem e-LICO[4].

3.1 TopBraid Composer

TopBraid Composer je komplexní nástroj pro modelování dat. Nabízí podporu pro vytváření, správu a testování konfigurací ontologií a RDF grafů. K dispozici je také vizuální editor RDF grafů a diagramů tříd, případně program obsahuje nástroje pro připojení k relačním databázím. Pro naši potřebu je však nejdůležitější funkcí možnost importování Excelovských sešitů, nebo obecně dat uložených v tabulkové struktuře do datového modelu RDF.

TopBraid Composer je schopný importovat tabulková data v několika různých formátech, například v klasických formátech Excelovských sešitů (*xls* a *xlsx*). Import tabulkových dat do modelu lze také provést pomocí textových formátů *txt* a *csv*, ve kterých jsou jednotlivé sloupce dat v tabulce odděleny tabulátory (TSV – Tab Separated Values), případně čárkami (CSV – Comma Separated Values). Jeden textový soubor pak představuje právě jednu tabulku.

V případě importování Excelovských sešitů je třeba připravit data do požadovaného formátu, kde jedna tabulka na jednom listu sešitu představuje třídu, hlavičky sloupců představují vlastnosti a v dalších buňkách sloupců jsou hodnoty vlastností. Instancemi třídy jsou jednotlivé řádky v tabulce. Příklad takové tabulky je znázorněn na obrázku 3.1. Tabulka představuje

[1] https://www.topquadrant.com/tools/modeling-topbraid-composer-standard-edition/

[2] https://www.topquadrant.com/company/

[3] http://e-lico.eu/populous.html

[4] http://e-lico.eu/#menu-e-lico

třídu osob se třemi instancemi a pěti vlastnostmi.

	A	B	C	D	E
1	jmeno	vek	datumNarozeni	pohlavi	pribuznyS
2	Bob	44	1973-08-16	Muz	Roman
3	Alice	41	1976-09-25	Zena	
4	Roman	14	2003-11-19	Muz	Bob

Obrázek 3.1: Příklad tabulky třídy osob pro import

Pokud však importujeme podobnou tabulku, vlastnosti jsou implicitně nastaveny jako datové. Chceme-li tak propojit dvě instance pomocí vlastnosti objektové, je třeba v uživatelském prostředí programu udělat další manuální úpravy, které jsou v našem případě nežádoucí a přidávají tak zbytečnou práci navíc. Jedná se také o komerční produkt, který má navíc mnoho dalších, pro náš problém nepotřebných, funkcionalit. Při převodu dat z Excelu do OWL program umožňuje vytvářet jedince včetně popisných tvrzení, ale nelze s ním definovat nové třídy a vlastnosti. Převod z OWL ontologie do Excelovského sešitu tento program nepodporuje, proto jako oboustranný převodník nevyhovuje.

3.2 Populous

Populous je nástroj pro sběr dat při vytváření ontologie. Uživatelské rozhraní Populous nabízí uživatelům tabulkový formulář, pomocí kterého můžeme prohlížet a vyplňovat data do již existujícího Excelovského sešitu, případně můžeme vytvořit sešit nový [6]. Jedná se o open source Java aplikaci s jednoduchým uživatelským rozhraním podobným programu Microsoft Excel. Obrázek rozhraní Populous se nachází v příloze.

Populous pomáhá ve fázi sběru znalostí při vývoji ontologií, odděluje sběr znalostí od konceptualizace a axiomizace. Není náhradou standardních nástrojů na správu ontologie, ale poskytuje jednoduchou platformu s cílem oslovit doménové experty během vytváření ontologií. [6]

Při vytváření nové ontologie pomocí Populous je potřeba datům v jednotlivých buňkách přiřadit ontologická omezení. K tomu ale potřebujeme naimportovat externí ontologie přes kontextové menu programu. Ontologie lze importovat z lokálního úložiště, nebo z BioPortalu[5] [7]. Hierarchie prvků importovaných ontologií jsou poté zobrazeny v pravé části uživatelského rozhraní.

[5]http://bioportal.bioontology.org/

Definování ontologických omezení na určitý rozsah buněk se provádí označením rozsahu buněk tabulky a zvolením prvku z importované ontologie. Příkladem ontologického omezení může být definice vztahu buněk z rozsahu k vybranému prvku importované ontologie, jedna z buněk může být například instancí daného prvku. Po aplikaci omezení Populous validuje zadané hodnoty v buňkách, případně našeptává při jejich vyplňování. [7]

Převod dat do OWL ontologie pomocí průvodce je poměrně komplikovaný a vyžaduje znalost skriptovacího jazyka OPPL[6]. Vzhledem k tomu tak Populous neodpovídá našim požadavkům, při vytváření nové samostatné ontologie z dat v Excelovském sešitu je tento postup nepoužitelný. Navíc se nejedná o oboustranný Excel - OWL převodník a rozšířit Populous podle našich potřeb taktéž nepřichází v úvahu. Při svém hledání jsem tak nenašel vyhovující řešení, v další kapitole se budu zabývat návrhem řešení vlastního převodníku.

[6]http://oppl2.sourceforge.net/

4 Návrh řešení

Obsahem této kapitoly je návrh řešení oboustranného Excel - OWL převodníku. Hlavním problémem bylo nalezení způsobu, jak strukturovat převedené informace a znalosti z OWL ontologie do Excelovského sešitu. V úvodu kapitoly se nachází sekce 4.1 s krátkým popisem situace před vytvořením převodníku, následována sekcí 4.2 se shrnutím požadavků na řešení převodníku. V další sekci 4.3 se nachází stručný návrh struktury zápisu ontologie do Excelu. Kapitola je zakončena návrhem workflow převodníku pro oba typy převodu.

4.1 Aktuální situace

V současné době je obtížné komunikovat se znalostními experty určité domény. Problém je především ve fázi modelování znalostní domény pomocí OWL ontologií a správě znalostí v nich uložených.

Doménoví experti z různých odvětví, kteří poskytují informace a znalosti při modelování ontologií, se mohou jen obtížně přímo podílet na vývoji ontologií. S technologií OWL totiž nejsou blíže seznámeni a nerozumí ji. Vytváření, či úprava zdrojového kódu OWL ontologie v libovolné syntaxi pomocí textového editoru kvůli tomu nepřichází v úvahu. Vzhledem k neznalostem s oblastí OWL není ideální ani práce v editorech, která může být i pro méně zkušeného člověka v této oblasti poměrně matoucí, jak jsem se o tom sám v začátcích přesvědčil.

Oproti tomu, základní práce s kancelářskými programy, jako je například tabulkový procesor Excel od firmy Microsoft[1], je expertům poměrně dobře známá. Tento fakt je hlavním důvodem toho, proč je převod orientován do tabulkové struktury Excelovských sešitů.

Tabulková struktura je ve své podstatě jednoduchá, přehledná a snadno upravitelná. Bude-li převod do tabulek správně navržen, neměl by mít obyčejný uživatel problém se ve struktuře orientovat a pracovat s ní. Další výhodou tohoto řešení je podle mého názoru skutečnost, že na většině pracovních počítačů je v dnešní době nainstalována sada kancelářských programů. V případě, že nemá uživatel přístup ke komerční variantě v podobě Excelu, je k dispozici alternativa v podobě nekomerčního balíku OpenOffice[2], která je

[1] https://www.microsoft.com/cs-cz

[2] https://www.openoffice.org/cs/

taktéž schopná se soubory tohoto typu pracovat.

Právě ze zmíněných důvodů vytvářím převodník, který odbourá technologickou bariéru v komunikaci s doménovými experty a umožní jim, aby se pomocí prostředí Excelu mohli přímo podílet na návrhu a modelování ontologií.

4.2 Požadavky na řešení

Cílem práce je naprogramovat převodník schopný převést platnou OWL ontologii do Excelovského sešitu, do kterého převodník strukturovaně zapíše informace a znalosti uložené v ontologii. Ze strukturovaných dat uložených v Excelovském sešitu převodník musí umět zpětně vytvořit validní OWL ontologii.

Pro uživatele, který bude převodník využívat, by měla být jeho obsluha co nejjednodušší. Uživatel předá při spuštění převodníku cestu k vstupnímu souboru a převodník vygeneruje výstup. Bližší popis workflow navrhovaného převodníku se nachází v sekci 4.4.

Při návrhu strukturalizace dat v Excelovském sešitu bylo třeba brát v potaz několik důležitých požadavků. Prvním z požadavků bylo, aby během konverze nedošlo ke ztrátě dat. Zachovány musí být všechny entity ontologie i jejich sémantika (třídní hierarchie, definice typů vlastností a podobně).

V zadání je také vymezeno, že by převodník měl přepsat data do Excelovského sešitu v takové struktuře, která bude přehledná a přívětivá pro doménové experty. Experti budou se strukturovanými daty v Excelovském sešitu pracovat při vývoji ontologií. Struktura by měla být navržena tak, aby uživateli umožňovala s daty manipulovat například pomocí filtrů, případně prohozením sloupců nebo řádků docílit lepší organizace dat na základě jeho preferencí.

Všechny tyto požadavky jsem bral v úvahu a pokusil jsem se navrhnout strukturu dat, která požadavky splňuje. Návrh struktury je stručně popsán v následující sekci 4.3.

4.3 Návrh struktury dat v Excelu

Kromě zohlednění zmíněných požadavků bylo při vymýšlení strukturalizace dat v Excelovském sešitu třeba vyhodnotit, zda bude lepší všechna data ontologie poskládat na jeden list, případně je rozdělit na více listů sešitu. Rozhodl jsem se data rozdělit na více listů, především kvůli lepší organizaci dat a přehlednosti pro cílového uživatele.

V prvním kroku bylo třeba se zamyslet nad tím, jaké prvky se v ontologii mohou vyskytovat. Na základě toho jsem ve svém návrhu struktury rozdělil celkem na čtyři listy. Každý list představuje určitou část ontologie, obsahující související množinu prvků. Prvním z listů sešitu je list *Ontology*, obsahující obecný popis ontologie. Každý z ostatních tří listů zachycuje entity jednoho typu, obsahem listu *Classes* je seznam tříd v ontologii, na listu *Properties* je popis vlastností v ontologii a tabulka na listu Individuals obsahuje všechny jedince v ontologii. Finální podoba struktury dat v Excelovském sešitu bude podrobněji popsána v sekci 5.1 následující kapitoly.

4.4 Workflow převodníku

Navrhovaná aplikace převodníku by měla načíst uživatelem předaný vstupní soubor, zpracovat data v něm uložená a v případě úspěšného převodu vygenerovat výstupní soubor. Typy vstupních a výstupních souborů, včetně postupu převodu, se budou lišit v závislosti na typu převodu. Interakce s uživatelem bude vyžadována pouze při zadávání vstupních parametrů, bez ohledu na typ převodu. O případných chybách v průběhu převodu by měl být uživatel informován chybovými výpisy na obrazovce. V následujících sekcích popisuji navrhovaný postup u obou typů převodu.

4.4.1 Převod z OWL do Excelu

Převod ontologie z OWL souboru do Excelu začne předáním cesty ke zdrojovému OWL souboru ontologie pomocí parametru programu. OWL soubor musí být v jedné z podporovaných syntaxí. Podporované syntaxe budou zmíněny ve fázi implementace. Následně dojde k načtení informací a znalostí ontologie skriptem do paměti. Na základě načtených dat bude na disk vygenerován výstupní Excelovský sešit.

Při analýze prvků obsažených v ontologii jsem kromě zmíněných prvků v sekci 4.3 narazil na komplexní prvky, které nebude možné jednoduše zapsat do Excelovské šablony. Navíc by tyto prvky mohly být matoucí pro doménové experty, kteří se nepohybují v oblasti OWL. Jedná se například o kolekce, nebo prázdné uzly. Dva příklady využití komplexních prvků jsou ve výpisu 4.1. V prvním případě se jedná o definici množiny disjunktních vlastností. Druhá část příkladu zachycuje definici omezení subjektu libovolné vlastnosti. V příkladu se vyskytují prázdné uzly a kolekce.

```
[ rdf:type owl:AllDisjointProperties ;
  owl:members ( nihss:datetimeExam
                nihss:patientID
                nihss:score
              )
] .

:propertyX rdfs:domain [ rdf:type owl:Class ;
                         owl:unionOf ( dcm:Series
                                       dcm:Study
                                       owl:Thing
                                     )
                       ] ;
```

Výpis 4.1: Příklad využití komplexních prvků v syntaxi Turtle, prázdné uzly jsou ohraničeny hranatými, kolekce kulatými závorkami

Po konzultaci s vedoucím práce jsem se rozhodl komplexní prvky při převodu do Excelovského sešitu oddělit. V případě, že se v převáděné ontologii tyto prvky vyskytují, budou umístěny do přídavného OWL souboru. V přídavném OWL souboru budou uložena tvrzení, ve kterých se komplexní prvky vyskytují. Výstupem převodu budou v takovém případě dva soubory.

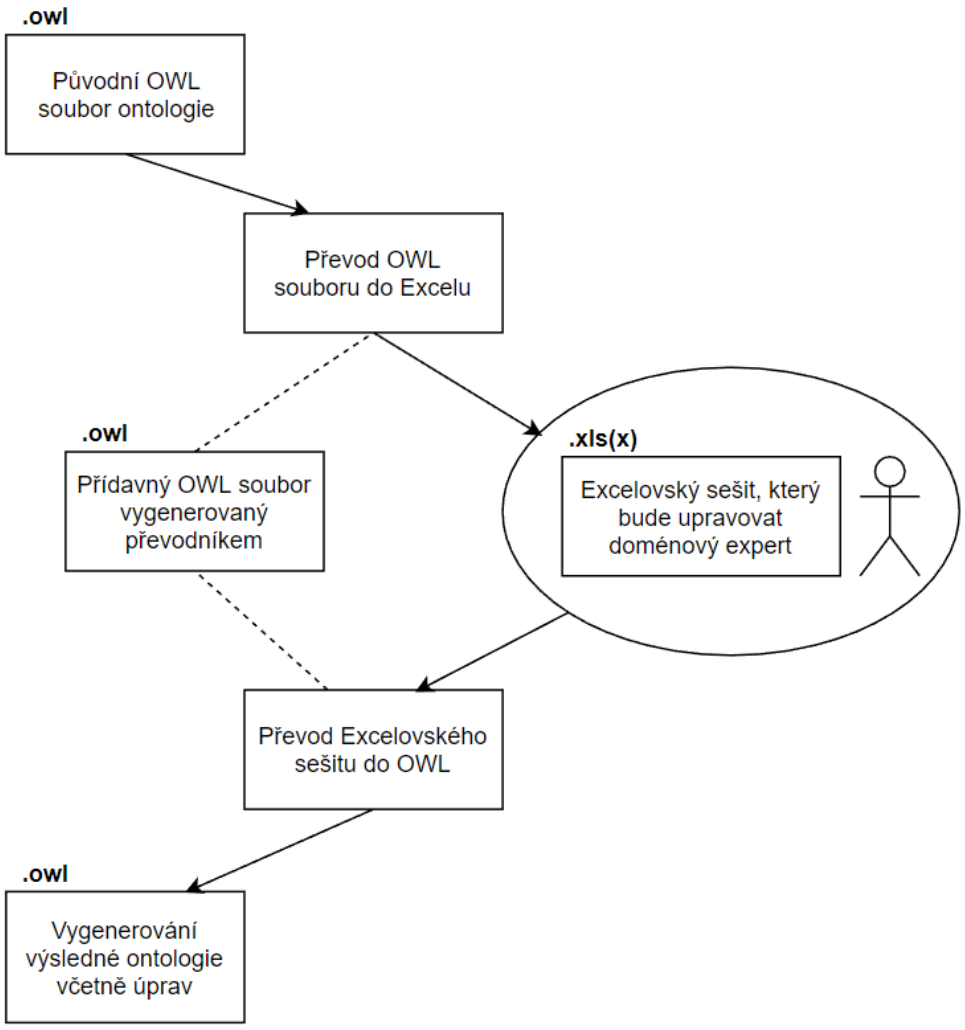
4.4.2 Převod z Excelu do OWL

Při převodu z Excelu do OWL mohou být vstupní soubory dva. Kromě Excelovského sešitu může vstupním souborem být ještě přídavný OWL soubor obsahující komplexní prvky zmíněné v sekci 4.4.1. Nejprve bude zpracován Excelovský soubor, data z něj budou skriptem načtena do paměti. V případě, že bude předán ještě přídavný OWL soubor, dojde k jeho zpracování. Posledním krokem bude vygenerování výsledného validního OWL souboru na základě zpracovaných dat z jednoho, nebo dvou vstupních souborů.

4.4.3 Využití převodníku

Na obrázku 4.1 je zobrazen navrhovaný postup při reálném využívání převodníku v průběhu modelování ontologie za pomoci doménového experta. Původní OWL soubor ontologie bude převeden pomocí převodníku do Excelovského sešitu. Během převodu do Excelu může vzniknout přídavný OWL soubor, pokud převáděná ontologie obsahuje komplexní prvky. Excelovský sešit bude k dispozici doménovému expertovi, který může provést změny v ontologii v prostředí Excelu.

Po provedení změn v Excelovském sešitu dojde ke zpětnému převodu do výsledného OWL souboru ontologie. Ve výsledném OWL souboru budou zachyceny provedené změny. Při zpětném převodu bude vstupem kromě Excelovského souboru přídavný OWL soubor, pokud vznikl při převodu ontologie do Excelu.



Obrázek 4.1: Postup při využívání převodníku

5 Implementace

Tato kapitola obsahuje popis implementace Excel - OWL převodníku. Programovacím jazykem, ve kterém jsem převodník implementoval, byla Java. Tento jazyk jsem zvolil především z důvodu, že s ním mám za svá studijní léta největší zkušenosti, navíc mi byl již při zadávání práce doporučen vedoucím práce.

Na začátku kapitoly popisuji způsob strukturalizace dat ontologie v Excelovském sešitu. V sekci 5.2 popisuji zvolené technologie, které jsem v průběhu implementace využíval. Další sekce 5.3 popisuje postup při implementaci převodníku a obsahuje popis jednotlivých tříd výsledného programu.

5.1 Struktura dat v Excelu

S přihlédnutím na požadavky zmíněné v předchozí kapitole jsem na začátku implementace vymyslel konkrétní podobu strukturalizace dat převáděné ontologie v Excelovském sešitu. Sešit je rozdělen na čtyři listy, kdy každý z nich zachycuje určitou část ontologie. Na prvním řádku tabulek všech listů je vždy hlavička popisující hodnoty na ostatních řádcích sloupce tabulky. Všechny listy jsou blíže popsány v následujících sekcích.

5.1.1 List Ontology

Prvním listem je list s názvem *Ontology*. Obsahem listu jsou základní informace o konvertované ontologii. V prvním sloupci tabulky listu se nachází URI ontologie. URI ontologie nemusí být nutně vyplněno. Je třeba jej vyplnit pouze v případě, že chceme do konvertované ontologie přidat importované ontologie, nebo přidat anotace ke konvertované ontologii (například jméno autora, nebo krátký popis obsahu ontologie). V opačném případě není nutné URI ontologie vyplňovat.

Druhý sloupec tabulky slouží k vyjmenování případných importovaných ontologií ve formě jejich URI. V následujících dvou sloupcích se nachází definice prefixů. V prvním z dvojice sloupců jsou prefixy ukončené znakem dvojtečky, v dalším sloupci je seznam URI prefixovaných jmenných prostorů. Využíváním prefixů si uživatel výrazně usnadní práci při vyplňování URI zdrojů do šablony. Bez prefixů by všechny zdroje musel uživatel do šablony zadávat v podobě kompletního URI.

Součástí prázdné šablony Excelovského sešitu jsou čtyři základní prefixy, které jsou využívány v rámci ostatních listů. Základní prefixy jsou zobrazeny v tabulce 5.1.

Prefix	**Prefix URI**
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
owl:	http://www.w3.org/2002/07/owl#
xsd:	http://www.w3.org/2001/XMLSchema#

Tabulka 5.1: Tabulka základních prefixů

V posledních dvou sloupcích tabulky jsou zaznamenány anotace vztahující se ke konvertované ontologii. Jedná se o seznam tvrzení, jejichž subjektem je konvertovaná ontologie. V prvním ze dvojice sloupců je název anotační vlastnosti (predikát tvrzení). Název vlastnosti může mít podobu URI vlastnosti v plném rozsahu, případně prefixovaného URI.

Ve druhém z dvojice posledních sloupců je ukládána hodnota anotace (objekt tvrzení). Je-li objektem tvrzení zdroj, do buňky s hodnotou stačí zapsat jeho URI v plném rozsahu, nebo v prefixované podobě. V případě více zdrojů na pozici objektu lze jejich URI oddělit čárkou a zapsat je do jedné buňky. Pokud je objektem tvrzení literál, za URI vlastnosti musí být v kulatých závorkách definováno URI datového typu literálu a v takovém případě lze uložit pouze jednu hodnotu. URI datového typu může být zadáno i v prefixované podobě.

U všech datových typů, kromě jednoho, stačí do závorek uvést pouze jeho URI. Výjimkou je řetězec s definovaným jazykem, u kterého je třeba za URI připojit kódové označení jazyka (pro angličtinu například „en“). URI datového typu a označení jazyka musí být v závorkách odděleno čárkou.

Zmíněná pravidla pro zápis vlastností na pozici predikátu i zdroje a literály na pozici objektu tvrzení platí v rámci všech listů datové struktury Excelovské šablony.

Na obrázku 5.1 je znázorněn příklad několika anotací libovolné ontologie v tabulce se zmíněnými prvky z předchozího textu. Všechny sloupce listu *Ontology* jsou statické, jejich pořadí je neměnné.

Pro zdroje, které jsou součástí jmenného prostoru prázdného prefixu platí, že mohou být kdekoliv v Excelu zapsány pouze ve tvaru lokálního jména zdroje. V takovém případě ale musí být prázdný prefix explicitně definován na listu *Ontology*.

	E	F
1	**Ontology Annotation**	**Annotation Value**
2	owl:versionIRI	http://example.com/2018/ontology.owl
3	owl:versionInfo (xsd:string)	2018/01
4	dc:issued (xsd:date)	2018-01-25
5	dc:contributor (xsd:string)	Franta Vomacka
6	dc:title (rdf:langString, cs)	Základní ontologie projektu XYZ
7	dc:title (rdf:langString, en)	XYZ Core Ontology

Obrázek 5.1: Příklad anotací ontologie v Excelu

5.1.2 List Classes

Obsahem listu *Classes* je seznam vyjmenovaných tříd v ontologii. Každý řádek tabulky listu představuje jednu entitu, v tomto případě se jedná o třídu. Všechny třídy na listu jsou instancí třídy `owl:Class`. První sloupec tabulky je statický a slouží pro uložení jména třídy. Do ostatních sloupců tabulky je možné ukládat tvrzení vztahující se k jednotlivým třídám, tyto sloupce jsou nestatické a lze v případě potřeby libovolně měnit jejich pořadí.

Chceme-li přidat tvrzení k jedné ze tříd, je třeba najít v hlavičkách sloupců název vlastnosti (predikátu tvrzení), včetně případného datového typu. V případě, že se v tabulce požadovaná hlavička nevyskytuje, název vlastnosti je třeba vyplnit do první prázdné hlavičky v tabulce. Hodnota (objekt tvrzení) bude uložena do buňky ve sloupci se jménem vlastnosti, na řádku odpovídajícím třídě v pozici subjektu tvrzení. Pro název vlastnosti na pozici predikátu a objekt tvrzení platí stejná pravidla jako pro anotace k ontologii.

Součástí listu *Classes* v prázdné Excelovské šabloně jsou tři základní vlastnosti. První z nich je `rdfs:subClassOf` pro definici třídní hierarchie. Další dva sloupce slouží k vyjádření ekvivalence, nebo disjunkce s jinými třídami pomocí vlastností `owl:equivalentClass` a `owl:disjointWith`.

5.1.3 List Properties

Na listu *Properties* se nachází definice a bližší charakteristika vlastností v ontologii. Každý řádek tabulky představuje jednu vlastnost a s ní související informace. První dva sloupce jsou statické. V prvním sloupci tabulky se nachází názvy definovaných vlastností. Ve druhém sloupci je uveden typ vlastnosti, tedy zda je vlastnost anotační, datová, nebo objektová, případně se jedná o obyčejnou vlastnost bez bližší specifikace typu.

Podobně jako ke třídám, i k vlastnostem lze přidat různá tvrzení. Přidání

tvrzení k vlastnostem se provádí stejně jako u tříd na listu *Classes* ve zbylých sloupcích tabulky a platí pro něj stejná pravidla.

I na listu *Properties* se nachází několik předvyplněných hlaviček pro bližší specifikaci vlastností. K definici oboru hodnot a definičního oboru vlastnosti slouží `rdfs:range` a `rdfs:domain`. Pro vyjádření hierarchie vlastností je v tabulce k dispozici sloupec s vlastností `rdfs:subPropertyOf`. Pomocí `owl:inverseOf` lze vlastnost označit za inverzní vůči jiné vlastnosti. Dalších sedm sloupců tabulky umožňuje vlastnost označit za funkční, případně ji modifikovat pomocí druhů binárních relací, například ji označit jako symetrickou, nebo reflexivní.

5.1.4 List Individuals

Obsahem listu *Individuals* je seznam jedinců. V prvním statickém sloupci se nachází názvy jedinců. Tvrzení k jedincům lze přidat stejně jako ke třídám, nebo vlastnostem na předchozích listech.

Součástí listu *Individuals* v prázdné šabloně je vlastnost `rdf:type`. Sloupec s touto vlastností slouží k vyjmenování tříd, jejichž instancí je jedinec na konkrétní řádce. Všichni jedinci jsou instancí třídy `owl:NamedIndividual`, tuto třídu proto není nutné explicitně do buňky zapisovat.

5.2 Zvolené technologie

Při implementaci převodníku jsem potřeboval efektivně pracovat s Excelovskými a OWL soubory. K práci s těmito soubory bylo třeba najít vyhovující Java knihovny. V prvním kroku jsem dal dohromady operace, které s jednotlivými typy souborů budu provádět. Hlavním kritériem bylo, aby vybraná knihovna byla schopna všechny tyto operace provést.

Co se týče práce s Excelovskými soubory, vybraná knihovna musela umět vytvořit čistý Excelovský soubor a přečíst již existující, ať už ve formátu *xlsx*, nebo starším *xls*. Knihovna musela být schopná pracovat s více listy sešitu. Při vytváření sešitu v podobě šablony zmíněné v sekci 5.1 musela knihovna umět naplnit jednotlivé listy sešitu požadovanými daty, případně provést jednoduché formátování buněk pro lepší přehlednost jednotlivých tabulek. Při čtení vstupního souboru bylo vyžadováno čtení dat na jednotlivých listech, bez ohledu na formátování buněk, nebo datový typ hodnot v nich uložených.

Všechny tyto požadavky splňovala knihovna Apache POI[1], navíc výhodou této knihovny je dobré zpracování dokumentace včetně mnoha prak-

[1] https://poi.apache.org/

tických příkladů pro jednotlivé operace na webových stránkách knihovny. Proto jsem si vybral právě tuto knihovnu, v sekci 5.2.1 ji blíže popisuji.

Při práci s OWL soubory musela knihovna umět načíst existující OWL soubor ontologie a vytvořit nový OWL soubor na základě zadávaných dat získaných z Excelu. Při čtení existujícího OWL souboru jsem po knihovně požadoval schopnost přístupu k jednotlivým prvkům ontologie popsaným u šablony strukturalizace dat v Excelu. Knihovna musela umět zpracovat například prefixy ontologie, anotace k ontologii, procházet jednotlivými entitami v ontologii (třídami, vlastnostmi a jedinci) a obecně pracovat se všemi tvrzeními v ontologii.

Pro implementaci převodníku s ohledem na práci s OWL soubory jsem si vybral knihovnu Apache Jena[2]. Při hledání řešení nějakého problému na internetu jsem povětšinou narazil na tuto knihovnu. Knihovna je používána na katedře a v projektech MRE[3], navíc mi byla doporučena vedoucím práce. Knihovně Apache Jena se blíže věnuji v sekci 5.2.2.

5.2.1 Apache POI

Cílem projektu Apache POI je vytvoření a údržba Java API pro manipulaci s různými formáty souborů založených buď na OOXML (Office Open XML) standardu (například soubory ve formátu *xlsx*, *docx*, *pptx*), nebo na OLE2 (Microsoft Compound Document File) formátu (například *xls*, *doc*, *ppt*). V projektu existuje několik API, z nichž každá se specializuje na dokumenty jedné z aplikací sady Microsoft Office [8]. Nejrozšířenější z nich jsou API na práci s Excelovskými sešity. Jedná se o POI-HSSF a POI-XSSF, které jsem během implementace převodníku používal.

POI-HSSF a POI-XSSF

HSSF (Horrible SpreadSheet Format) a XSSF (XML SpreadSheet Format) API umožňují Excelovské sešity vytvářet, číst a modifikovat. Implementace HSSF slouží k práci se sešity ve starším formátu *xls*, zatímco implementace XSSF dokáže pracovat se sešity ve formátu *xlsx* (Excel verze 2007 a novější) [9]. Kombinované rozhraní SS sdružuje funkcionalitu HSSF a XSSF a umožňuje práci s oběma formáty zároveň, proto jsem během implementace převodníku využíval převážně toto rozhraní.

Pro reprezentaci Excelovských sešitů se v rozhraní SS využívá objektů třídy `Workbook`, ať už se jedná o formát *xls* (HSSFWorkbook), nebo *xlsx*

[2]https://jena.apache.org/
[3]https://mre.zcu.cz/

(XSSFWorkbook). Pomocí objektu `Workbook` lze například vytvářet nové listy sešitu, případně získat přístup k existujícím. Jednotlivé listy jsou reprezentovány objektem `Sheet`, který umožňuje například změnu šířky sloupců, nebo ukotvení určité části tabulky listu.

Přístup k buňkám listu je realizován kombinací objektů `Row` a `Cell`. Inicializací objektu `Row` lze vybrat libovolný řádek listu, pomocí objektu `Cell` lze následně přistupovat k jednotlivým buňkám vybraného řádku. Prostřednictvím objektu `Cell` lze číst, případně upravovat hodnoty v buňkách. Hodnoty buněk mohou být reprezentovány jako obyčejný řetězec znaků, vzorec, nebo numerická hodnota (například buňky formátovány jako číslo, datum, nebo čas). Formátování buněk se provádí s využitím třídy `CellStyle` nad jednotlivými buňkami, nebo řádky.

5.2.2 Apache Jena

Apache Jena je framework, který poskytuje API umožňující práci s RDF, OWL, nebo použití dotazovacího jazyka SPARQL nad RDF daty. Při implementaci převodníku jsem využil převážně funkcionalitu RDF API[4] a Jena Ontology API[5] pro práci s jednotlivými prvky ontologií.

Pro reprezentaci ontologií v paměti programu se v Apache Jena využívá tzv. modelů, objektů třídy `Model`, případně `OntModel`. Komplexnější `OntModel` má oproti základnímu modelu některé další funkcionality pro zjednodušení práce s ontologiemi, například získání iterátoru nad třídami, nebo vlastnostmi v ontologii. Jena také nabízí tři operace mezi instancemi třídy `Model`. Využitím dvou modelů lze vytvořit třetí model sjednocením, průnikem, nebo rozdílem množiny tvrzení původních dvou. [10]

Jena umožňuje přistupovat k jednotlivým trojicím v modelu. V takovém případě jsou trojice reprezentovány instancemi třídy `Statement`. Nové trojice lze do modelu přidávat, stejně tak je možné existující trojice z modelu odstranit. V trojicích se mohou vyskytovat zdroje (reprezentovány třídou `Resource`), nebo literály (reprezentovány třídou `Literal`). Na pozici objektu trojic může být zdroj i literál, proto je pro manipulaci s objekty trojic využíváno rozhraní `RDFNode`, které implementují obě třídy `Resource` a `Literal`. Zdroje jsou v rámci modelu identifikovány svým URI. Instance literálů obsahují hodnotu a mají definované URI datového typu. [10]

Při převodu z OWL do Excelu bylo třeba načíst existující ontologii do modelu, zatímco v případě opačného převodu jsem model postupně vytvářel na základě informací uložených v Excelovském sešitu.

[4] https://jena.apache.org/documentation/rdf/

[5] https://jena.apache.org/documentation/ontology/

5.3 Implementace aplikace převodníku

Samotnou implementaci aplikace Excel - OWL převodníku jsem prováděl v několika fázích. V první fázi bylo potřeba naučit se v Javě pracovat s Excelovskými sešity s využitím Apache POI, od čtení hodnot z existujících souborů, po vytváření nových souborů a formátování tabulek. Inspirací mi byly především dobře zpracované praktické příklady na webových stránkách POI-HSSF a POI-XSSF API[6]. Následně jsem byl schopný vytvořit prázdnou Excelovskou šablonu popsanou v sekci 5.1.

V další fázi jsem se učil v Javě zpracovávat OWL soubory ontologií pomocí Apache Jena, s využitím objektů rozhraní `Model` a `OntModel`. Zkoušel jsem přistupovat k jednotlivým entitám zpracovávaných ontologií, abych byl schopný je přepsat do Excelovského sešitu. Zkoušel jsem také vytvářet úplně novou ontologii, včetně všech prvků obsažených v Excelovské šabloně. V této fázi jsem hojně využíval Java dokumentaci Apache Jena[7], ve které jsou popsány jednotlivé funkcionality použitých rozhraní.

Po tom, co jsem si osvojil základní manipulaci s oběma typy souborů a jejich obsahem, jsem se pustil do implementace jednodušší verze převodníku. Prvotní verzi převodníku jsem testoval využitím mnou vytvořené jednoduché ontologie, která obsahovala alespoň jednou všechny prvky z Excelovské šablony. Postupem času jsem funkcionalitu převodníku rozšiřoval a v pozdních fázích implementace jsem již využíval ontologie projektu MRE.

Ve finální verzi má převodník podobu konzolové Java aplikace. Převodník je schopný převést OWL ontologii do Excelovského sešitu ve struktuře popsané v sekci 5.1. Převod do Excelu probíhá podle popisu ze sekce 4.4.1 v předchozí kapitole. Při opačném převodu z Excelu do OWL byl také dodržen navrhovaný postup v sekci 4.4.1. V případě potřeby aplikace umožnuje vygenerování prázdné Excelovské šablony.

O tom, která operace bude provedena, rozhodne uživatel zadáním parametrů při spuštění aplikace. Výpisu návodu k použití aplikace lze docílit spuštěním aplikace bez parametrů. Vstupní soubory musí být zadány absolutní cestou, nebo relativní cestou vzhledem k umístění aplikace. U výstupních souborů je požadováno pouze jméno, po jejich vygenerování jsou umístěny do předem definované složky. Vstupní Excelovské soubory mohou být buď ve formátu *xls*, nebo *xlsx*. OWL soubory musí být v jedné z podporovaných syntaxí. Podporovanými syntaxemi jsou RDF/XML a Turtle.

Obsahem následujících sekcí je popis jednotlivých tříd implementace Excel - OWL převodníku. Výsledná aplikace převodníku se skládá ze čtyř tříd.

[6] https://poi.apache.org/spreadsheet/quick-guide.html

[7] https://jena.apache.org/documentation/javadoc/jena/index.html

První třída `TemplateCreation` zprostředkovává vytvoření prázdné Excelovské šablony. Dalšími dvěma třídami jsou `OWLReading` a `ExcelReading`, každá z těchto tříd provádí jeden typ převodu. Poslední třídou je třída `Converter`, která je hlavní třídou programu.

5.3.1 Třída TemplateCreation

Třída `TemplateCreation` obsahuje funkcionalitu na vytváření prázdné Excelovské šablony pro zápis dat při převodu ontologie z OWL do Excelu. Třída je využívána i v případě, že uživatel požaduje pouhé vygenerování prázdné šablony. Umožněno je vytvoření šablony ve formátu *xls* i *xlsx*.

Součástí třídy je také formátování jednotlivých listů šablony, například ukotvení prvního řádku tabulek, formátování hlaviček, nastavení šířky všech sloupců na základě vyplněných hodnot, nebo nastavení validace hodnot pro buňky s očekávaným obsahem. Pro validaci buněk je využíváno rozhraní `DataValidation`. Validací hodnot jsem se snažil omezit chyby uživatele při upravování šablony.

Ve třídě jsou definovány veřejné konstanty pro práci se šablonou, využívané i ostatními třídami. Jedná se například o jména listů šablony, která jsou pevně definována.

5.3.2 Třída OWLReading

Třída `OWLReading` zprostředkovává převod ontologie ze vstupního OWL souboru do Excelovského sešitu. Metoda `convertOWLToExcel` provádí proces převodu, v průběhu využívá ostatních metod ve třídě. Nejprve je načten vstupní OWL soubor do objektu `OntModel`. V případě, že vstupní soubor nebyl nalezen, je program ukončen s chybou.

U načítání OWL souborů jsem narazil na chybu v implementaci Apache Jena verze 3.6.0, kde jedna z variant metody `read` v rozhraní `Model` nefungovala tak, jak bylo popsáno v dokumentaci. Při načítání souboru pomocí URL metoda ignorovala druhý parametr v podobě řetězce se zadanou syntaxí, proto pro načítání OWL souborů využívám variantu s parametrem `InputStream`.

Podaří-li se vstupní soubor načíst, je zpracován jeho obsah a proveden zápis do Excelovské šablony. Zpracování obsahu ontologie je prováděno v několika krocích. Nejdříve jsou zpracována a zapsána data na list *Ontology*, počínaje prefixy. Následně jsou zpracovány případné anotace k ontologii a jsou zaznamenány URI importovaných ontologií.

Po zpracování obecných informací o ontologii a jejich zápisu na list *Ontology* následuje zpracování tříd, vlastností a jedinců v ontologii, společně s tvrzeními ve kterých jsou tyto prvky obsaženy. K získání řetězcové reprezentace zdrojů pro zápis na jednotlivé listy Excelovského sešitu slouží metoda `getResourceString`. Metoda `getPropertyHeader` vrací řetězcovou reprezentaci pro predikát tvrzení, zapisovaného na listech sešitu. Reprezentace predikátu obsahuje i datový typ, je-li objektem tvrzení literál. Při zápisu literálů do Excelu je využívána lexikální forma literálů v podobě řetězce. Zápis tvrzení do sešitu provádí metoda `addItemTriple`.

Při načtení ontologie do objektu `OntModel`, jsou do objektu načtena i tvrzení z importovaných ontologií, proto bylo třeba při zpracování tříd, vlastností a jedinců rozlišovat, zda jsou součástí převáděné ontologie, nebo importované. Do Excelovské šablony jsou zaznamenány pouze entity z převáděné ontologie. Pro zachování prvků z importovaných ontologií stačí zaznamenat URI importované ontologie na listu *Ontology*.

Po zápisu všech potřebných informací do Excelovského sešitu v paměti následuje jeho vytvoření na disku do složky „ExcelOWL converter - Excel output“. Složka je vytvořena ve stejném adresáři, ve kterém se nachází aplikace převodníku. Do této složky je také umístěn přídavný OWL soubor zmiňovaný v sekci 4.4.1. Vytvoření přídavného OWL souboru je poslední funkcionalitou třídy.

5.3.3 Třída ExcelReading

Třída `ExcelReading` obsahuje funkcionalitu pro převod z Excelu do OWL. Třída obsahuje metodu `convertExcelToOWL`, která provádí proces převodu s využitím ostatních metod třídy. Převod začíná načtením vstupního Excelovského souboru do objektu `Workbook`. Nepodaří-li se nalézt vstupní soubor, je program ukončen s chybou.

V případě úspěšného načtení vstupního souboru je vytvořen prázdný objekt `OntModel`. Následuje zpracování jednotlivých listů v Excelovském sešitu. Zpracovávaná data jsou průběžně přidávána do modelu ontologie. Nejprve je zpracován list *Ontology* obsahující obecné informace o ontologii. Následně jsou zpracovány listy obsahující informace o třídách, vlastnostech a jedincích.

Při zpracovávání informací v sešitě jsou kontrolovány některé chyby uživatele, který sešit upravoval. Například na listu *Ontology* je kontrolováno chybějící URI u definovaného prefixu, nebo hodnota u anotace ontologie. Na ostatních listech je kontrolováno zadání jména entity v prvním sloupci. Kontrolováno je také formátování buňky obsahující hodnotu literálu. Formátování by mělo odpovídat datovému typu, nebo by měl být literál zadaný

v lexikální formě v podobě řetězce. O všech zjištěných chybách je uživatel informován výpisem do konzole.

Po zpracování informací ze sešitu následuje zpracování případného dodatečného OWL souboru. Tvrzení z dodatečného souboru jsou přidána do objektu `OntModel`. Posledním krokem je vytvoření výsledného OWL souboru ontologie, který je umístěn do složky „ExcelOWL converter - OWL output“. Složka je vytvořena ve stejném adresáři, ve kterém se nachází aplikace převodníku. Výstupní OWL soubor je v RDF/XML syntaxi. Je třeba brát na vědomí, že Jena ve verzi 3.6.0 při výpisu „opomíná“ prázdný prefix.

5.3.4 Třída Converter

Třída `Converter` je hlavní třídou programu, obsahuje hlavní metodu `main`. Ve třídě jsou zpracovávány vstupní parametry zadané při spuštění aplikace uživatelem a je ověřováno jejich správné zadání. Na základě zadaných parametrů třída zavolá funkcionalitu ostatních tříd za účelem provedení požadované úlohy, případně vypíše jednoduchý návod k použití aplikace v anglickém jazyce.

6 Diskuze výsledku

Jedním z bodů zadání bylo, abych ověřil funkčnost převodníku a provedl zhodnocení a diskuzi nad dosaženými výsledky. V zadání bylo stanoveno, že k tomu mám využít ontologie projektu MRE. Ontologie projektu MRE jsem využíval již v průběhu implementace převodníku.

6.1 Ověření funkčnosti

Pro ověření správné funkčnosti převodníku jsem vybral čtyři ontologie projektu MRE, zobrazené v tabulce 6.1. Počet tvrzení v ontologii jsem získal s pomocí Apache Jena, při načtení ontologie do objektu `Model` a získáním návratové hodnoty metody `size`. Poslední sloupec tabulky udává, zda byl při převodu ontologie vytvořen přídavný OWL soubor, obsahující tvrzení s komplexními prvky, které nebylo možné převést do Excelovského sešitu. Ontologie jsou v tabulce seřazeny vzestupně podle počtu tvrzení, všechny ontologie byly staženy z webových stránek projektu MRE[1].

Ontologie	**Počet tvrzení**	**Přídavný OWL soubor**
nihss.owl	937	Ano
ibd.owl	2 539	Ne
ibdt.owl	14 219	Ne
dcm.owl	34 124	Ano

Tabulka 6.1: Srovnávací tabulka použitých ontologií

Při ověřování funkčnosti převodníku jsem vybrané ontologie nejprve převedl do Excelovských sešitů v obou formátech (*xls* i *xlsx*), následně jsem provedl zpětný převod z obou formátů Excelu do OWL, včetně přídavných souborů. V tabulce 6.2 jsou zobrazeny časy jednotlivých převodů v milisekundách.

Pomocí metody `nanoTime` jsem získal časy běhů programu. Metoda je ve třídě `System`[2], která je součástí základního balíku Javy. Ihned po spuštění programu jsem zaznamenal první časovou stopu, druhou stopu jsem zaznamenal těsně před ukončením programu, čas běhu programu je rozdílem stop. Součástí časů běhu jsou i operace čtení z disku a zápis.

[1] https://mre.zcu.cz/ontology/ontologies.html

[2] https://docs.oracle.com/javase/8/docs/api/java/lang/System.html

Ontologie	Př. do Excelu (xls/xlsx)	Př. do OWL (xls/xlsx)
nihss.owl	1 740 ms / 3 213 ms	1 135 ms / 1 568 ms
ibd.owl	5 439 ms / 6 701 ms	1 239 ms / 1 786 ms
ibdt.owl	7 583 ms / 11 201 ms	1 793 ms / 2 800 ms
dcm.owl	9 280 ms / 17 782 ms	2 936 ms / 4 673 ms

Tabulka 6.2: Srovnávací tabulka časů převodu (včetně zápisu a čtení z disku)

Všechny výsledné časy zobrazené v tabulce 6.2 jsou aritmetickým průměrem pěti časů spuštění programu pro danou úlohu. Program jsem spouštěl na počítači s operačním systémem Microsoft Windows 10 Pro (64 bitová verze), s procesorem Intel Core i5 4200M s frekvencí 2,5GHz a operační pamětí DDR3 o velikosti 8 GB.

Po provedení operací převodu pro zmíněné ontologie a změření výsledných časů bylo třeba nějakým způsobem ověřit správnou funkčnost převodníku. V první řadě jsem musel ověřit, zda je výsledný OWL soubor ontologie validní. K tomu jsem využil editor ontologií Protege ve verzi 5.2.0. Všechny výstupní OWL soubory editor bez problémů zpracoval, s validitou výstupních OWL souborů tedy nebyl problém.

Hlavním kritériem implementace převodníku bylo zamezení ztráty dat z hlediska obsahu i sémantiky dat v průběhu převodu. Ke zjištění případné ztráty dat jsem porovnal původní OWL soubory převáděných ontologií s výstupními OWL soubory. Výstupní OWL soubory byly výsledkem převodu z Excelu zpět do OWL. K porovnání obsahu původního a výsledného OWL souboru jsem využil funkcionalit Apache Jena. Oba OWL soubory jsem načetl do objektu `Model`, následně jsem použil metodu `difference` z rozhraní `Model` pro vytvoření „rozdílového“ modelu.

Rozdílový model obsahoval pouze tvrzení, která se nevyskytují v obou modelech. Teoreticky by v případě neztrátového převodu měl být rozdílový model prázdný, ale u ontologií *nihss* a *dcm* tomu tak nebylo. Zapříčiněno to bylo tím, že obě ontologie obsahovaly komplexní prvky zmíněné v sekci 4.4.1. Tyto prvky nemají žádný pevný identifikátor a při každém načtení ontologie do modelu pomocí Jeny je prvkům přidělen jiný (náhodný) identifikátor, bylo tedy třeba tvrzení obsahující komplexní prvky zkontrolovat manuálně.

Po manuální kontrole jsem zjistil, že všechna tvrzení z původního OWL souboru jsou i ve výsledném OWL souboru. Během převodu tedy nedošlo k žádné ztrátě a hlavní kritérium převodníku bylo splněno.

6.2 Zhodnocení výsledků

V předchozí sekci byla ověřena správná funkčnost převodníku. S ohledem na časové hodnoty v tabulce 6.2 lze říci, že doba běhu programu u obou konverzí je ovlivněna především rozsahem převáděné ontologie, kdy s vyšším počtem tvrzení v převáděné ontologii stoupá čas potřebný na provedení převodu. Pracuje se s poměrně velkým množstvím dat, navíc převod je činnost prováděná sporadicky, proto jsou časy podle mého názoru přijatelné. Doménový expert se s časovým omezením nesetká, bude pracovat až v prostředí Excelu s převedenou ontologií.

Zajímavým zjištěním pro mě byl fakt, že převod při použití formátu *xlsx* Excelovského sešitu trvá znatelně delší dobu, než při použití staršího formátu *xls*. Platí to u obou typů převodu. Pravděpodobně je to zapříčiněno implementací frameworku Apache POI pro práci s Excelovskými sešity. V implementaci převodníku jsem totiž využíval univerzálního rozhraní SS, zmíněného v sekci 5.2.1, pro práci s oběma formáty Excelovských sešitů. Implementace pro oba formáty se tak v rámci převodníku lišila minimálně.

Z tabulky 6.2 je také patrné, že převod z OWL do Excelu je časově mnohem náročnější než převod z Excelu do OWL. Při převodu do Excelovského sešitu je totiž model ontologie několikrát procházen při zpracování různých kategorií dat obsažených v ontologii. Dochází také k vytváření šablony Excelovského sešitu, včetně zápisu zpracovávaných dat a formátování, které je v porovnání s vytvářením modelu v případě druhého typu převodu časově náročnější.

Práce s přídavným OWL souborem není tolik časově náročná. Během převodu do Excelu se jednou kompletně projdou tvrzení ontologie a jsou zachycena pouze tvrzení obsahující komplexní prvky. V případě převodu do OWL jsou tvrzení z přídavného OWL souboru přidána do modelu výsledné ontologie před vygenerováním výstupního OWL souboru. Časová náročnost těchto operací je vzhledem k ostatním operacím v průběhu převodu relativně zanedbatelná.

Výsledná implementace převodníku splňuje požadavky zmíněné v kapitole 4 z pohledu funkčnosti výsledné aplikace. Excelovské sešity vygenerované převodníkem lze upravovat pomocí aplikace Microsoft Excel, případně lze využít aplikace Calc z nekomerční sady Open Office.

Převodník je obousměrný, zvládá oba typy převodu mezi OWL a Excelem. Aplikace převodníku je jednoduchá na obsluhu, vyžaduje pouze zadání vstupních parametrů a v průběhu převodu již žádnou interakci od uživatele nevyžaduje. Aplikaci může uživatel využít v případě, že chce upravovat data v ontologii, nebo přidat do ontologie nové entity. V takovém případě uživatel

převede ontologii do Excelovské šablony, ve které provede změny a z šablony potom může zpětně vygenerovat pozměněný OWL soubor.

Excelovskou šablonu jsem se snažil navrhnout tak, aby byla pro uživatele přívětivá, přehledná a jednoduchá na použití při úpravě dat v ní uložených. Data v Excelovské šabloně jsem rozdělil na listy obsahující související části ontologie. Struktura všech listů je skoro stejná, včetně pravidel pro zápis jednotlivých prvků. Uživatel by se v ní proto měl bez větších problémů orientovat.

Porovnání mnou vytvořeného řešení Excel - OWL převodníku s existujícím řešením jiných autorů není možné. Jak již bylo zmíněno v kapitole 3, žádné řešení, které by funkcionalitou odpovídalo řešenému problému, jsem nenašel.

Jako možné rozšíření aplikace bych navrhoval vytvoření grafického uživatelského rozhraní. Dalším možným rozšířením by mohlo být zpracování komplexních prvků a jejich strukturovaný zápis do Excelovské šablony tak, aby vstupem i výstupem převodníku byl vždy právě jeden soubor a nebyl využíván žádný přídavný OWL soubor.

7 Závěr

Tématem této bakalářské práce bylo vytvoření aplikace převodníku mezi Excelovskými a OWL soubory. V počátku práce bylo cílem se seznámit s oblastí RDF a OWL. Tato část zadání je vypracována v kapitolách 1 a 2, ve kterých jsem se snažil o stručný nástin oblastí RDF a OWL, včetně souvisejícího RDF Schema. Popsal a vysvětlil jsem zde základní pojmy pro práci s OWL ontologiemi a datovým modelem RDF.

Po teoretickém základu bylo cílem práce najít podobná existující řešení zadaného převodníku. Při svém hledání jsem nenarazil na řešení, která by vyhovovala požadavkům. Tato část práce byla popsána v kapitole 3.

V další části práce bylo za úkol vymyslet způsob přepisu dat uložených v OWL ontologiích do Excelovského sešitu v přehledné a pro uživatele přívětivé struktuře. V kapitole 4 jsem se věnoval požadavkům na navrhovanou Excelovskou šablonu pro zápis dat z ontologií. Podrobnou podobu šablony jsem popsal v sekci 5.1.

Dalším bodem byla implementace zadaného Excel - OWL převodníku. Převodník jsem implementoval v programovacím jazyce Java s využitím externích frameworků Apache POI a Apache Jena pro práci s Excelovskými a OWL soubory. Použité frameworky byly popsány v sekci 5.2. Převodník se mi podařilo úspěšně implementovat, výstupem převodníku jsou OWL soubory a Excelovské soubory ve formátu *xls*, nebo *xlsx*.

Po implementaci převodníku jsem měl za úkol ověřit funkčnost převodníku a provést diskuzi a zhodnocení dosažených výsledků. Pro ověření funkčnosti jsem využil reálné ontologie projektu MRE. Ověření funkčnosti proběhlo úspěšně, při převodu vybraních ontologií jsem nezaznamenal žádné ztráty dat. V diskuzi jsem se věnoval výsledkům práce. Výsledky práce jsem okomentoval s ohledem na splnění definovaných požadavků v rámci návrhu řešení práce.

Pro splnění posledního bodu zadání jsem měl porovnat vlastní řešení převodníku s řešením jiných autorů, žádné podobné řešení jsem však nenašel. V příloze práce jsem vypracoval uživatelskou příručku pro uživatele, který bude aplikaci převodníku využívat.

Literatura

[1] *RDF 1.1 Primer* [online]. W3C, 2014. [cit. 2017/12/11]. Dostupné z: http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/.

[2] *Linked Data - Design Issues* [online]. 2010. [cit. 2018/06/01]. Dostupné z: https://www.w3.org/DesignIssues/LinkedData.html.

[3] DAVIS, M. – SUIGNARD, M. *Unicode Security Considerations - Visual Security Issues* [online]. 2010. [cit. 2018/01/20]. Dostupné z: http://unicode.org/reports/tr36/tr36-8.html#visual_spoofing.

[4] *RDF Schema 1.1* [online]. W3C, 2014. [cit. 2017/12/17]. Dostupné z: https://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[5] *Web Ontology Language Primer (Second Edition)* [online]. W3C, 2012. [cit. 2017/12/18]. Dostupné z: http://www.w3.org/TR/2012/REC-owl2-primer-20121211/.

[6] JUPP, S. et al. Populous: a tool for building OWL ontologies from templates. *BMC Bioinformatics*. 2012, 13, Suppl 1, s. S5.

[7] JUPP, S. et al. Populous: A tool for populating Templates for OWL ontologies. *arXiv preprint arXiv:1012.1745*. 2010.

[8] *Apache POI - the Java API for Microsoft Documents* [online]. The Apache Software Foundation, 2018. [cit. 2018/06/15]. Dostupné z: https://poi.apache.org/index.html.

[9] *POI-HSSF and POI-XSSF - Java API To Access Microsoft Excel Format Files* [online]. The Apache Software Foundation, 2018. [cit. 2018/06/15]. Dostupné z: https://poi.apache.org/spreadsheet/index.html.

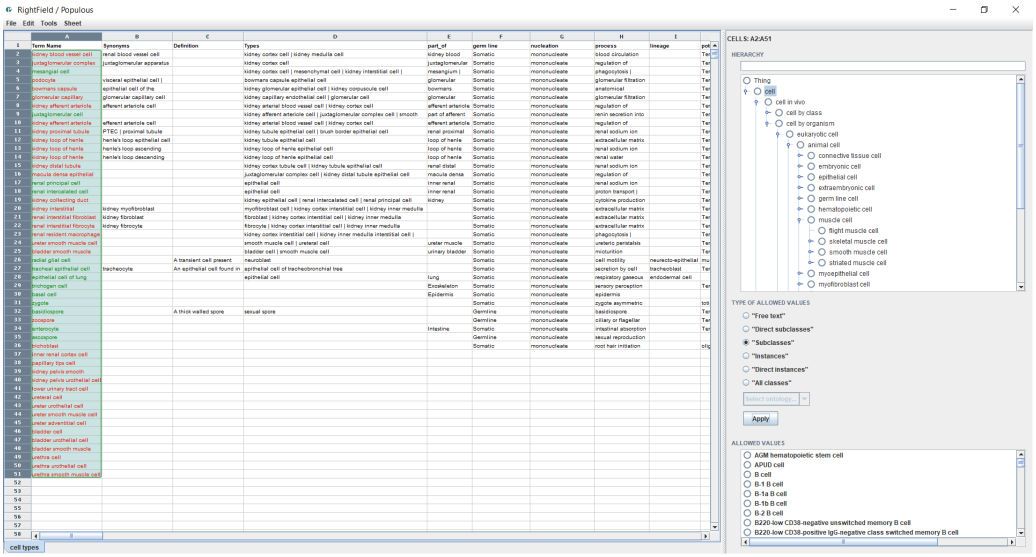
[10] *Apache Jena - An Introduction to RDF and the Jena RDF API* [online]. The Apache Software Foundation, 2018. [cit. 2018/06/15]. Dostupné z: https://poi.apache.org/spreadsheet/index.html.

Přehled zkratek

API - Application Programming Interface
ASCII - American Standard Code for Information Interchange
CSS - Cascading Style Sheet
CSV - Comma Separated Values
HTML - HyperText Markup Language
HSSF - Horrible Spreadsheet Format
IRI - Internationalized Resource Identifier
MRE - Medical Research and Education
OLE2 - Microsoft Compound Document File
OOXML - Office Open XML
OPPL - Ontology PreProcessing Language
OWL - Web Ontology Language
RDF - Resource Description Framework
SPARQL - Simple Protocol and RDF Query Language
TSV - Tab Separated Values
URI - Uniform Resource Identifier
URL - Uniform Resource Locator
W3C - World Wide Web Consortium
XML - Extensible Markup Language
XSSF - XML SpreadSheet Format

Přílohy

Obrázky



Obrázek 1: Rozhraní nástroje Populous, podobné rozhraní aplikace Microsoft Excel, umožňuje přímo upravovat Excelovské soubory. V pravém horním rohu se nachází hierarchie tříd načtené ontologie. Pod hierarchií je možné definovat obsah buněk vybraného rozsahu v tabulce. Je možné například označit oblast v tabulce a označit ji za množinu jedinců zvolené třídy ve třídní hierarchii. Do buněk tabulky lze vložit více hodnot oddělených znakem „|“. Oddělení hodnot se hodí například u přiřazení typu k jedincům.

Obsah přiloženého DVD

Na přiloženém DVD jsou následující složky:

doc – obsahuje .pdf soubor s textem bakalářské práce.

doc-src – obsahuje zdrojové soubory textu bakalářské práce, včetně všech použitých obrázků v podsložce **img**.

lib – tato složka obsahuje využívané knihovny pro práci s OWL soubory (Apache Jena, verze 3.6.0) a pro práci s Excelovskými soubory (Apache POI, verze 3.17).

src – obsahuje všechny zdrojové soubory Java aplikace převodníku.

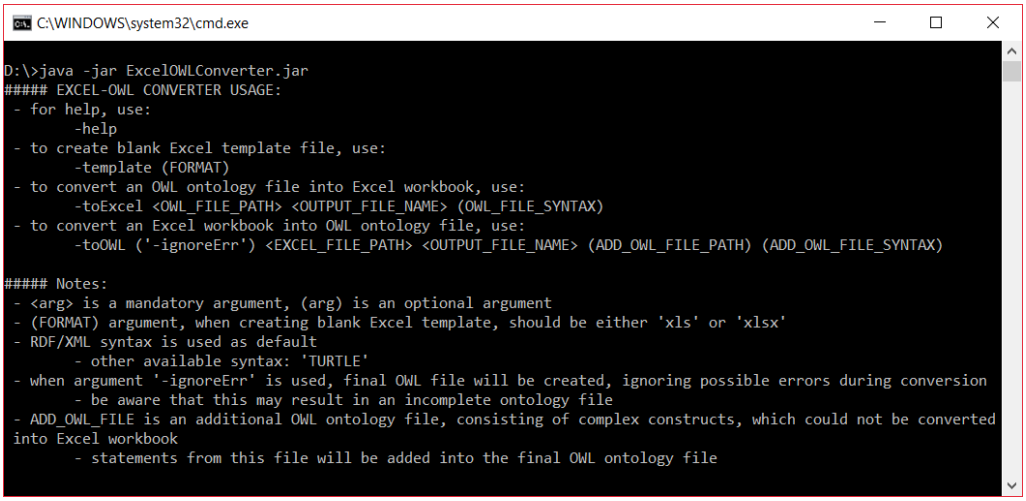
out – ve složce se nachází spustitelná .jar aplikace převodníku. Mimo aplikaci jsou součástí složky skripty pro ukázkový běh aplikace. Skript „run.bat“ je určen pro spuštění na operačním systému Microsoft Windows, skript „run.sh“ je určen pro spuštění na běžných distribucích Linuxu (například Ubuntu, nebo Debian). Skripty využívají ukázkové soubory v podsložce **data**. Pro správnou funkčnost skriptů a samotné aplikace je třeba celý obsah složky **out** z DVD vykopírovat do nějakého adresáře, ve kterém má uživatel právo zápisu.

Uživatelská příručka

Program převodníku je konstruován tak, aby byl funkční na všech počítačích s operačním systémem Microsoft Windows 7, 8 a 10 v 64 bitové verzi. Program lze spustit i na běžných distribucích Linuxu (například Ubuntu, nebo Debian). Jedná se o konzolovou aplikaci a pro její úspěšné spuštění je nutné mít na počítači nainstalovanou Javu verze 8 a novější.

Spuštění programu

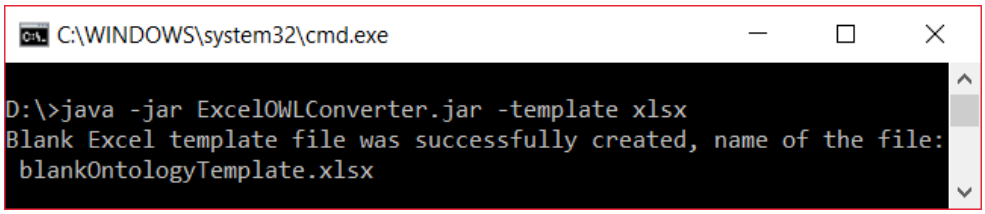
Pro spuštění programu je nutné v terminálu libovolného OS vstoupit do adresáře, ve kterém se nachází spustitelný soubor aplikace převodníku. Poté příkazem `java -jar ExcelOWLConverter.jar` spustíme program, v tomto případě bez parametrů. Při spuštění programu bez parametrů se vypíše jednoduchý návod k použití v anglickém jazyce (viz obrázek 2).



Obrázek 2: Spuštění programu bez parametrů

Vygenerování prázdné Excelovské šablony

Chce-li uživatel pomocí převodníku vygenerovat prázdnou Excelovskou šablonu, například při vytváření nové ontologie, je třeba program spustit s povinným prvním parametrem `-template`. Další parametr je nepovinný, jedná se o formát souboru výsledné šablony. Uživatel si může vybrat z formátu *xls* a *xlsx*, při nezadání parametru je zvolen výchozí formát *xls*. Výstupní Excelovské soubory je možné upravovat aplikací Excel ze sady Microsoft Office, případně nekomerční aplikací Calc ze sady Open Office. Ukázka vygenerování prázdné šablony je na obrázku 3.

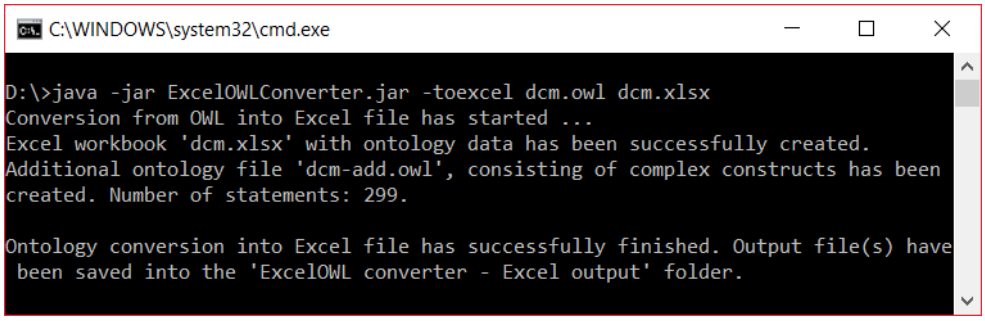


Obrázek 3: Vygenerování prázdné šablony

Převod z OWL do Excelu

Pro převod OWL souboru do Excelu je třeba zadat první povinný parametr `-toExcel`. Druhý i třetí parametr je také povinný. Druhým parametrem musí být absolutní cesta ke zdrojovému OWL souboru na disku, případně lze použít i cestu relativní vzhledem k pracovnímu adresáři. Třetím parametrem je jméno výstupního souboru včetně přípony, která udává formát výstupního souboru (*xls* nebo *xlsx*). Jméno výstupního souboru si uživatel může zvolit.

Posledním (nepovinným) čtvrtým parametrem je určení syntaxe zdrojového OWL souboru. Převodník je schopný zpracovat OWL soubory v syntaxi RDF/XML a Turtle. V případě, že není OWL soubor v syntaxi RDF/XML, je třeba do čtvrtého parametru zadat název syntaxe vstupního souboru. Výstupem převodu OWL souboru do Excelu může být kromě Excelovského souboru ještě přídavný OWL soubor, obsahující tvrzení zachycující komplexní prvky, které není převodník schopný zapsat do Excelovského sešitu. Příklad převodu OWL ontologie do Excelu pomocí převodníku je na obrázku 4.



Obrázek 4: Převod OWL ontologie do Excelu

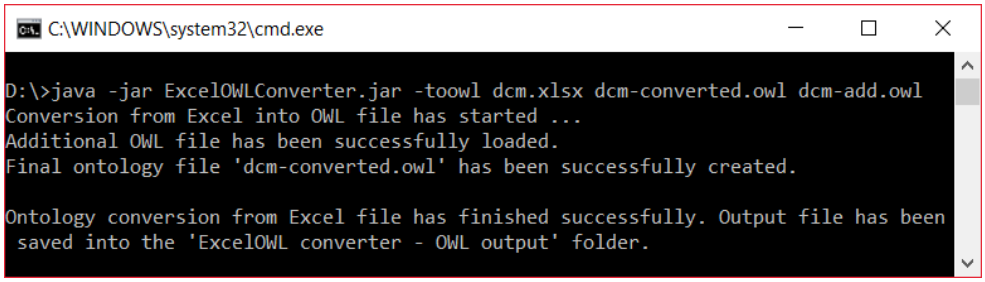
Převod z Excelu do OWL

Pro převod z Excelu do OWL je třeba zadat první povinný parametr `-toOWL`. Druhý parametr `-ignoreErr` je nepovinný. Lze ho použít v případě, že chce uživatel získat výstupní OWL soubor i přes případné zjištěné chyby ve zdrojovém Excelovském souboru. To může vést k vygenerování nekompletního

výstupního OWL souboru. Není-li tento parametr zadán, je výstupní OWL soubor vygenerován pouze v případě bezchybného převodu.

Třetí a čtvrtý parametr je povinný. Třetím parametrem je absolutní cesta ke zdrojovému Excelovskému souboru na disku, případně lze použít cestu relativní vzhledem k pracovnímu adresáři. Vstupní Excelovský soubor musí být v podobě definované šablony a musí být ve formátu *xls*, případně *xlsx*. Čtvrtým parametrem je jméno výstupního OWL souboru včetně přípony. Jméno výstupního souboru si může uživatel zvolit.

Pátý a šestý parametr není povinný. Pátým parametrem je cesta k přídavnému OWL souboru. Tvrzení obsažena v tomto souboru jsou připojena do výsledného OWL souboru. Šestý parametr slouží pro definování názvu syntaxe přídavného OWL souboru, není-li soubor v syntaxi RDF/XML. Výstupem úspěšného převodu je jeden OWL soubor v syntaxi RDF/XML s daty ze zdrojového Excelovského souboru a přídavného OWL souboru, je-li tento soubor zadán. Příklad převodu do OWL je na obrázku 5.



Obrázek 5: Převod z Excelu do OWL

Podporované datové typy

Při vyplňování hodnot literálů v Excelovské šabloně musí být dodržena určitá pravidla s ohledem na datový typ literálu. Datový typ literálu musí být zadán do závorek za název vlastnosti na pozici predikátu v podobě URI (v prefixovaném, nebo plném rozsahu). Výjimkou je řetězec s definovaným jazykem, jazykové označení se přidává do závorky za URI a je od URI odděleno čárkou.

Při převodu z ontologie do Excelovského sešitu jsou všechny hodnoty, bez ohledu na datový typ, zapsány v řetězcové reprezentaci. Při manuálním vyplňování dat je možné u některých datových typů použít jednoduché formátování buněk. V následujících odstavcích budou popsány datové typy, u nichž je možné formátování použít.

Celé číslo (např. xsd:int, xsd:integer) Buňku je možné formátovat jako číslo, případně je možné je zapsat v řetězcové reprezentaci.

Desetinné číslo (např. xsd:double, xsd:decimal) Buňku je možné formátovat jako desetinné číslo, případně použít řetězcovou reprezentaci s desetinným oddělovačem v podobě znaku čárky, nebo tečky.

Pravdivostní hodnota (např. xsd:boolean) Je možné použít řetězcovou reprezentaci („true“, nebo „false“), případně lze využít formátování buňky pro pravdivostní hodnoty.

Datum (např. xsd:date) Buňku s hodnotou datumu je možné formátovat jako datum. Druhou možností je zadat datum v podobě řetězcové reprezentace `YYYY-MM-DD`.

Čas (např. xsd:time) Buňka s časovou hodnotou musí být formátována jako čas, případně je možné využít řetězcové reprezentace v podobě `hh:mm:ss`.

Ostatní datové typy musí být zadány v jejich řetězcové reprezentaci. Příklad využití datových typů je na obrázku 6. Prefixy využité v této sekci jsou součástí Excelovské šablony.

	A	B	C	D
1	**Property Name**	**Property Type**	**fnmId (xsd:int)**	**rdfs:label (rdf:langString, cs)**
2	age	Data	8	věk
3	allergy	Data	19	alergie
4	anamnesisDailyActivities	Object	68	omezení denní aktivity
5	anamnesisDate	Data	999	datum
6	anamnesisDeterioration	Data	73	zhoršení stavu
7	anamnesisDiet	Object	75	strava
8	anamnesisDietText	Data	999	strava – text

Obrázek 6: Využití datových typů při přiřazení hodnot literálů